Table 3: Parameters used for generating training materials: prompts, the LLM model used, and the corresponding temperature.

Material	Generation goal	Model	T
Test case category hint	To generate test categories that are meaningful for the programming exercise.	GPT-3.5	0.3
	`[Sys.]` You are a helpful and experienced teaching assistant of an introductory programming class. `[User]` Problem Description: {problem_description}. List three most important aspects of this problem that need to be tested by describing the type of input. Write only each aspect in 3-6 words `[LLM ]` {A list of test category names}		
Test case desc. hint	To accurately describe the key aspect a test case is covering.	GPT-3.5	0.1
	`[Sys.]` You are a helpful and experienced teaching assistant of an introductory programming class. `[User]` Problem Description: {problem_description}. Briefly describe this test case's input and explain what important aspect of this problem that the following test case covers: {test_case} `[LLM ]` {A description of the key aspect} `[User]` Reformat it as a one-sentence hint. Use this template: Write a test case to cover the scenario where ... `[LLM ]` {One sentence hint in the right format}		
Buggy code	To over-generate bugs with mixed quality for selecting behaviorally distinct practice codes.	GPT-3.5	0.7

	`[Sys.]` You are a **novice student in intro CS**, you make mistakes and write buggy code when solving a problem.
	`[User]` ### Problem Description: {problem_description}
	### Instruction: Write **different buggy solutions** with common mistakes like novice students.
	### Buggy Implementations:
	`[LLM ]` {buggy code #1}
	`[LLM ]` {buggy code #2}
	`[LLM ]` {...}		

Bug explanation and fix instruction	To describe each unique bug, and write a corresponding fix instruction. If there are multiple bugs in the code, generate their explanations and fixes separately.	GPT-4	0.3
	`[Sys.]` You are a helpful and experienced **teaching assistant** of an introductory programming class. `[User]` Hi, I'm a student in your class. I'm having trouble with this problem in the programming assignment: {problem_description} I've tried to fix my code but I'm still stuck. Can you help me? `[LLM ]` Sure, let's take a look at your code. `[User]` Here's my buggy code: {buggy_code} What's wrong with my code? **List all the unique bugs included, but do not make up bugs**. For each point, put in the format of: {explanation: accurate and concise explanation of what the code does and what the bug is, for a novice, fix: how to fix the bug, within 30 words} Only **return the bullet list**. Do not write any other text or code. `[LLM ]` {Bullet list of json formats with explanation and fixes}		
Bug fix	To edit the buggy code according to the fix instruction, without over- or under- fix.	GPT-3.5	0.3
	`[Sys.]` You fix bugs in Python code closely following the instructions. `[User]` Original code: {buggy_code}; Code modification: {explanation} **Translate the statement into actual, minimal code change in this format**: {original code snippet: ""copy the lines of code that need editing"" -> edited code snippet: ""write the edited code snippet""} `[LLM ]` {JSON formatted old to new snippet, e.g., numbers_list[i] <= key → numbers_list[i] > key } `[Sys.]` You fix bugs in Python code closely following the instructions. `[User]` Old Code:{buggy_code}; Instruction:{Old snippet to new snippet}; New Code: `[LLM ]` {The complete version of the new fixed code}		

- *Test case category hint*: A good test case category should be *reasonable to test for a problem*. We define the success rate as the proportion of *suitable test categories* over all the test categories per problem.
- *Test case description hint*: A good test case hint should accurately *describe the test case behavior*. We define the success rate as the proportion of test case hints that provide an accurate description.
- *Buggy codes*: Intuitively, we prefer buggy codes that add more meaningful variations to existing buggy code collections. Therefore, we define success rate as *the ratio of behaviorally distinct codes*, where code behavior is automatically categorized with instructor-provided reference test inputs.
- *Bug explanation and fix instruction*: As the prompt requires identifying multiple bugs, we manually evaluate each pair of explanation and fix instructions, and count one generation as correct when there is a matching explanation and the corresponding fix, and both are correct.[6]
- *Bug fix*: We define a bug fix to be a success if the { old code snippet – > new code snippet } accurately reflects the instruction without over- or under- fix.

[6]In some cases the LLM describes the same bug twice because there are conceptual differences between them even when they actually share the same fix. We still count this as correct, as they only require humans to delete the extra bugs.

Table 4: Description, number of generated bugs, and number of final fixed codes for problems used in LLM experiment

Problem	Description	# Bugs	# Fixes
first_num_greater_than	`Write a Python function first_num_greater_than(numbers_list, key) that takes a list of integers (numbers_list) and an integer key (key), and returns the first number in the list that is greater than the key. If there is no number greater than the key, then you should return None.`	30	36
remove_extras	`Function remove_extras(lst) takes in a list of integers and returns a new list with the first occurrence of each element, which is the same as lst but with all repeated occurrences of any element removed.`	23	31
num_smaller	`Function num_smaller(seq, x) takes in an integer x and a sorted integer sequence seq, and returns the number of elements in seq that is strictly smaller than x.`	18	24
sort_age	`We represent a person using a tuple (<gender>, <age>). Given a list of people, write a function sort_age that returns a list in an order such that the older people are at the front of the list. You may assume that no two members in the list of people are of the same age.`	28	42
top_k	`Write a function top_k that takes a list of integers as the input and returns the greatest k number of values as a list, with its elements sorted in descending order. You may use any sorting algorithm, but you are not allowed to use the Python function sort and sorted.`	28	42
swap_keys_values	`Write a function swap_keys_values that takes in a dictionary, and returns a new dictionary with the keys and values swapped.`	18	20
Total		145	195

Table 5: Human annotation tasks and success rate definitions for each type of material generated in LLM experiment

Material	Success Rate Definition	Human Label Tasks
Buggy code	$\frac{\text{\# behaviorally distinct buggy code}}{\text{\# non-identical buggy code}}$	remove extra comments e.g., `# the operator should actually be >`
Bug explanation and fix instruction	$\frac{\text{\# correct expl \& fix}}{\text{\# total buggy code}}$	`multibug_correct:` `T/F` (does this code have meaningful and isolate fixes specify for the bug(s) listed), `NA` (there is only one single bug as specified) `expl_correct:` `T/F` (does the explanation accurately reflect the bug in code) `fix_correct:` `T/F` (does the fix correctly and minimally fix the bug in code) `edited_expl_fix`: edited for next step if needed `{expl:...; fix:...}`
Fixed Code	$\frac{\text{\# accurately fixed code}}{\text{\# total buggy code}}$	`fix_round1_correct:` `T/F` (does the code snippet accurately reflect the instruction without over- or under-fix) & `edited_round1_fix`: edited fix lines if needed `fix_round2_correct:` `T/F` (does the fixed code accurately reflect the instruction without over or under-fix) & `edited_fixed_code`: edited fix code if needed
Test case description hint	$\frac{\text{\# correct test case hint}}{\text{\# total test case}}$	`tc_hint_correct:` `T/F` (does this hint accurately describe what the test case is doing) & `edited_tc_hint`: edited test case hint if needed
Test case category	$\frac{\text{\# correct test category}}{\text{\# total test category}}$	`tc_category_correct:` `T/F` (is this category reasonable to test for this problem) & `edited_tc_category`: edited category if needed